基于opencv如何实现视频中的颜色识别功能

本篇内容介绍了“基于opencv如何实现视频中的颜色识别功能”的有关知识，在实际案例的操作过程中，不少人都会遇到这样的困境，接下来就让小编带领大家学习一下如何处理这些情况吧！希望大家仔细阅读，能够学有所成！

**颜色识别的原理**

**opencv中的颜色模型**

RGB

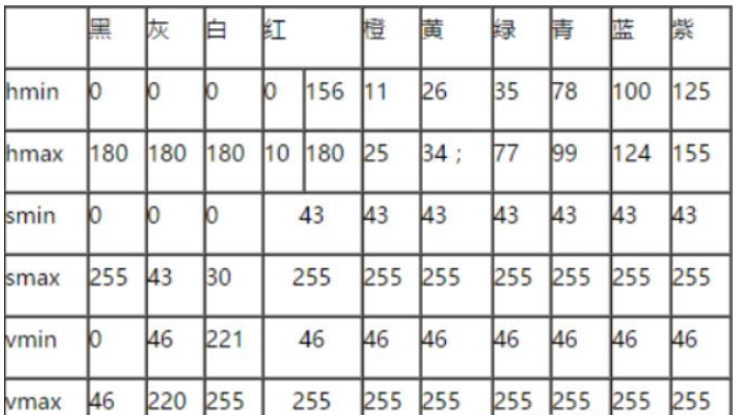
RGB具有三个通道其，分别表示红色通道&reg;，绿色通道(G)，蓝色通道(B)，3个通道在opencv中的取值均为0~255，它的颜色由3个通道的取值来共同决定，因此如果使用RGB图像来进行颜色的识别，会丢失很多的颜色。

HSV

HSV具有三个通道，其分别表示色调(H)，饱和度(S)，亮度(V），3个通道在opencv中的取值分别如下：

* H:0~180
* S:0~255
* V:0~255

其中H通道和S通道决定了颜色，V通道决定亮度  
各种颜色对应的三个通道的取值表如下：



因此通过限制HSV通道中相对应的数值，就可以识别出对应的颜色

**颜色识别的实现（c++）**

1.读取摄像头的实时画面

VideoCapture capture(1);//0为电脑本身摄像头，1位外置摄像头

2.读取摄像头的当前一帧的数据并转换到HSV空间

capture >> frame; //读取当前帧

cvtColor(frame, imgHSV, COLOR\_BGR2HSV);

3.对HSV图像进行直方图均衡化  
在此处使用直方图均衡化是因为可以使用这种方法可以使原来比较少像素的灰度会被分配到别的灰度去，像素相对集中， 处理后灰度范围变大，对比度变大，清晰度变大，所以能有效增强图像。

split(imgHSV,temporary);

equalizeHist(temporary[2], temporary[2]);

merge(temporary, imgHSV);   //将HSV图像分割通道，并且做直方图的均衡化

使用equalizeHist（）函数，其输入必须是单通道的，因此使用split函数将得到的HSV图像分割为三个通道之后，在进行直方图均衡化，然后再使用merge函数来合并三个通道。  
此处仅对V通道进行了直方图均衡化，因为通过比对发现只对V通道进行效果最好，具体的原因还不清楚。

4.将直方图均衡化之后，使用inRange(）函数来进行图像的识别

inRange(imgHSV, Scalar(H\_W\_L, S\_W\_L, V\_W\_L), Scalar(H\_W\_H, S\_W\_H, V\_W\_H), image);

其中H\_W\_L,H\_W\_L, S\_W\_L, V\_W\_L. . . 等数值分别为前面表格对应的颜色值的最小值和最大值。  
此函数是将在范围内的像素值为255，其与为0，反映到图像上就是，选中的颜色为白色，其与的均为黑色。  
操作到此颜色可以进行识别，但是效果可能不会很好，因此可以再使用开操作来消除噪点，去掉小的干扰快，再使用闭操作来填充闭合区域。

5.开操作

kernel = getStructuringElement(MORPH\_RECT, Size(3, 3));//3\*3的矩形卷积核，只要是奇数的都可以

morphologyEx(image,image,2,kernel);

6.闭操作

morphologyEx(image,image,3,kernel);

源代码

#include <opencv2\opencv.hpp>

using namespace cv;

using namespace std;

int main()

{

Mat frame,imgHSV,image,kernel;

vector<Mat>temporary;

int H\_W\_L = 0, H\_W\_H = 180;

int S\_W\_L = 0, S\_W\_H = 30;

int V\_W\_L = 221, V\_W\_H = 255;

VideoCapture capture(1);//读取视摄像头实时画面数据，0默认是笔记本的摄像头；如果是外接摄像头，这里改为1

while (true)

{

capture >> frame; //读取当前帧

if (!frame.empty())

{          //判断输入的视频帧是否为空的

cvtColor(frame, imgHSV, COLOR\_BGR2HSV);

//threshold(gray, two\_value, 125, 255, THRESH\_BINARY);//二值图像

split(imgHSV,temporary);

equalizeHist(temporary[2], temporary[2]);

/\*equalizeHist(temporary[1], temporary[1]);

equalizeHist(temporary[0], temporary[0]);\*/

merge(temporary, imgHSV);   //将HSV图像分割通道，并且做直方图的均衡化

inRange(imgHSV, Scalar(H\_W\_L, S\_W\_L, V\_W\_L), Scalar(H\_W\_H, S\_W\_H, V\_W\_H), image);

kernel = getStructuringElement(MORPH\_RECT, Size(3, 3));

morphologyEx(image,image,2,kernel);

morphologyEx(image,image,3,kernel);

imshow("直方图", imgHSV);

imshow("原图",frame);

imshow("window", image);  //在window窗口显示frame摄像头数据画面

}

if (waitKey(20) == 'q')   //延时20ms,获取用户是否按键的情况，如果按下q，会推出程序

break;

}

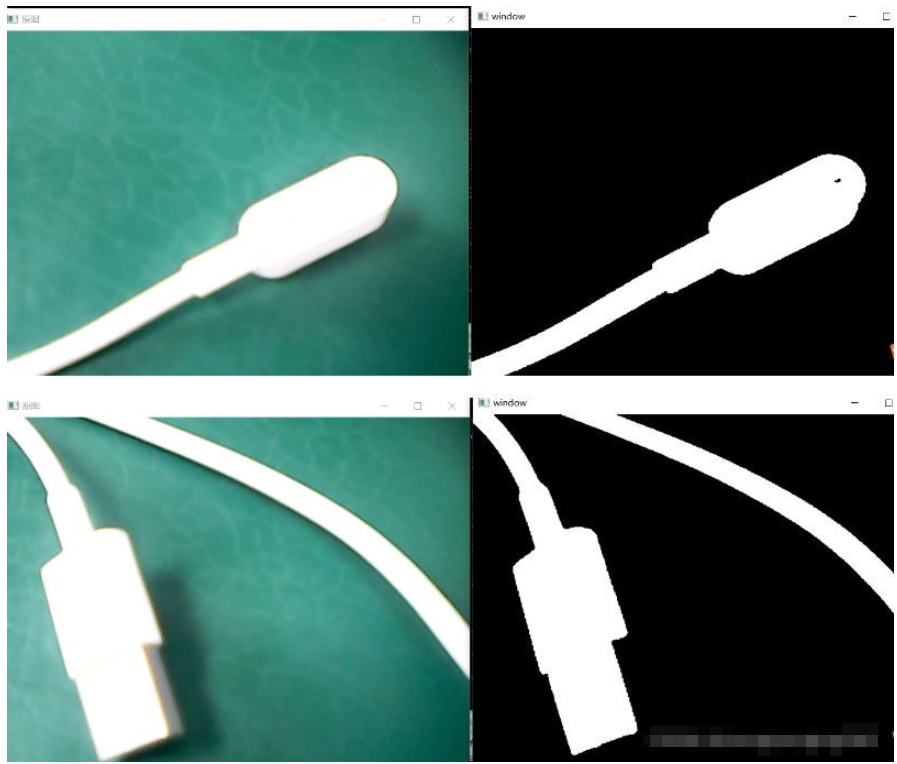
capture.release();     //释放摄像头资源

destroyAllWindows();   //释放全部窗口

return 0;

}

##结果



“基于opencv如何实现视频中的颜色识别功能”的内容就介绍到这里了，感谢大家的阅读。如果想了解更多行业相关的知识可以关注亿速云网站，小编将为大家输出更多高质量的实用文章！

HSV颜色识别\_opencv颜色识别原理

[0](https://javajgs.com/archives/150501#respond)

HSV颜色识别\_opencv颜色识别原理出处：百度文库一般对颜色空间的图像进行有效处理都是在HSV空间进行的，然后对于基本色中对应的HSV分量需要给定一个严格的范围，下面是通过实验计算的模糊范围（准确的范围在网上都没有给出）。H:0—180S:0—255V:0—255此处把部分红色归为紫色范围：目前在计算机视觉领域存在着较多类型的颜色空间(c...

一般对颜色空间的图像进行有效处理都是在HSV空间进行的，然后对于基本色中对应的HSV分量需要给定一个严格的范围，下面是通过实验计算的模糊范围（准确的范围在网上都没有给出）。

H:  0 — 180

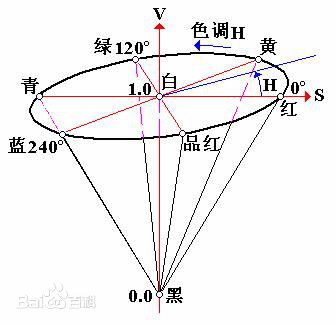
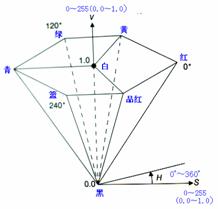
S:  0 — 255

V:  0 — 255

此处把部分红色归为紫色范围：







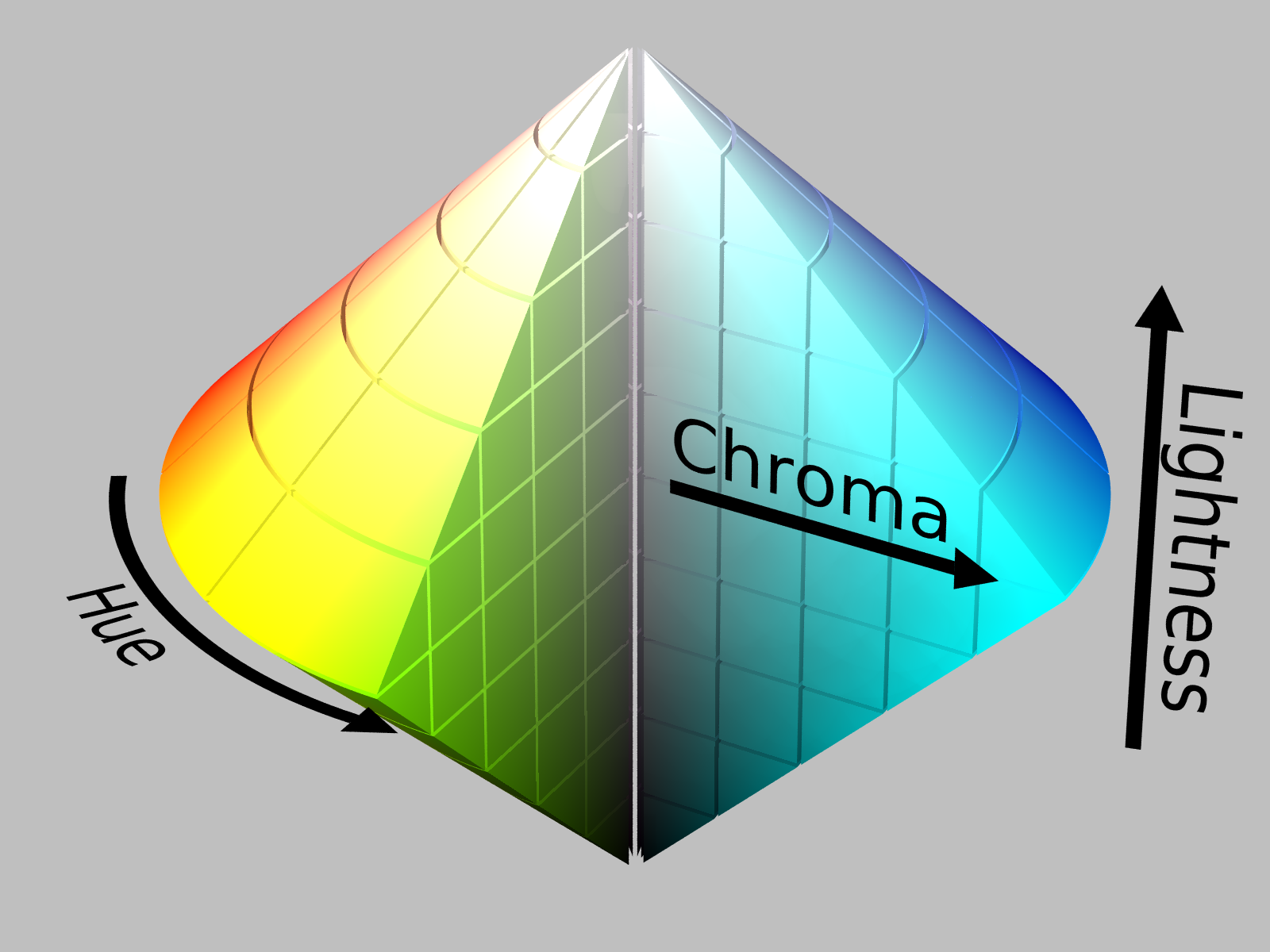
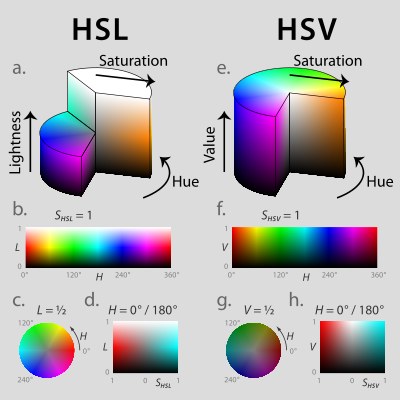
目前在计算机视觉领域存在着较多类型的颜色空间(color space)。HSL和HSV是两种最常见的圆柱坐标表示的颜色模型，它重新影射了RGB模型，从而能够视觉上比RGB模型更具有视觉直观性。

**HSV颜色空间**  
HSV(hue,saturation,value)颜色空间的模型对应于圆柱坐标系中的一个圆锥形子集，圆锥的顶面对应于V=1. 它包含RGB模型中的R=1，G=1，B=1 三个面，所代表的颜色较亮。色彩H由绕V轴的旋转角给定。红色对应于 角度0° ，绿色对应于角度120°，蓝色对应于角度240°。在HSV颜色模型中，每一种颜色和它的补色相差180° 。 饱和度S取值从0到1，所以圆锥顶面的半径为１。HSV颜色模型所代表的颜色域是CIE色度图的一个子集，这个 模型中饱和度为百分之百的颜色，其纯度一般小于百分之百。在圆锥的顶点(即原点)处，V=0,H和S无定义， 代表黑色。圆锥的顶面中心处S=0，V=1,H无定义，代表白色。从该点到原点代表亮度渐暗的灰色，即具有不同 灰度的灰色。对于这些点，S=0,H的值无定义。可以说，HSV模型中的V轴对应于RGB颜色空间中的主对角线。 在圆锥顶面的圆周上的颜色，V=1，S=1,这种颜色是纯色。HSV模型对应于画家配色的方法。画家用改变色浓和 色深的方法从某种纯色获得不同色调的颜色，在一种纯色中加入白色以改变色浓，加入黑色以改变色深，同时 加入不同比例的白色，黑色即可获得各种不同的色调。

**HSI颜色空间**  
HSI色彩空间是从人的视觉系统出发，用色调(Hue)、色饱和度(Saturation或Chroma)和亮度 (Intensity或Brightness)来描述色彩。HSI色彩空间可以用一个圆锥空间模型来描述。用这种 描述HIS色彩空间的圆锥模型相当复杂，但确能把色调、亮度和色饱和度的变化情形表现得很清楚。 通常把色调和饱和度通称为色度，用来表示颜色的类别与深浅程度。由于人的视觉对亮度的敏感 程度远强于对颜色浓淡的敏感程度，为了便于色彩处理和识别，人的视觉系统经常采用HSI色彩空间， 它比RGB色彩空间更符合人的视觉特性。在图像处理和计算机视觉中大量算法都可在HSI色彩空间中 方便地使用，它们可以分开处理而且是相互独立的。因此，在HSI色彩空间可以大大简化图像分析 和处理的工作量。HSI色彩空间和RGB色彩空间只是同一物理量的不同表示法，因而它们之间存在着 转换关系。

HSI 色彩模型是从人的视觉系统出发，用 H 代表色相 (Hue)、S 代表饱和度 (Saturation) 和 I 代表亮度 (Intensity) 来描述色彩。饱和度与颜色的白光光量刚好成反比，它可以说是一个颜色鲜明与否的指标。因此如果我们在显示器上使用 HIS 模型来处理图像，将能得到较为逼真的效果。   
色相 (Hue)：指物体传导或反射的波长。更常见的是以颜色如红色，橘色或绿色来辨识，取 0 到 360 度的数值来衡量。   
饱和度 (Saturation)：又称色度，是指色彩的强度或纯度。饱和度代表灰色与色调的比例，并以 0% (灰色) 到 100% (完全饱和) 来衡量。   
亮度 (Intensity)：是指颜色的相对明暗度，通常以 0% (黑色) 到 100% (白色) 的百分比来衡量。

HSL代表色调(Hue)，饱和度(Saturation)和亮度(Lightness)，通常也称为HLS。HSV代表色调，饱和度和值(Value)。注意HSL 和HSV的两个H的含义是相同的，而饱和度的定义是不同的，虽然都叫饱和度，从后面的定义可以看出二者的不同。  
HSL 和 HSV 二者都把颜色描述在圆柱体内的点，这个圆柱的中心轴取值为自底部的黑色到顶部的白色而在它们中间是的灰色，绕这个轴的角度对应于“色相”，到这个轴的距离对应于“饱和度”，而沿着这个轴的距离对应于“亮度”，“色调”或“明度”。 HSV 以人类更熟悉的方式封装了关于颜色的信息：“这是什么颜色？深浅如何？明暗如何？”。HSL 颜色空间类似于 HSV，在某些方面甚至比它还好。HSL的模型为双圆锥形状。  
这两种表示在用目的上类似，但在方法上有区别。二者在数学上都是圆柱，但 HSV（色相，饱和度，明度）在概念上可以被认为是颜色的倒圆锥体（黑点在下顶点，白色在上底面圆心），HSL 在概念上表示了一个双圆锥体和圆球体（白色在上顶点，黑色在下顶点，最大横切面的圆心是半程灰色）。HSV 模型在 1978 年由埃尔维?雷?史密斯创立。下图给出了HSL和HSV的圆柱模型。



在圆锥上，角度代表色调H，饱和度S表示为点到中心竖线的距离，而亮度或者值V用中心竖线表示。红色的角度为0度，依次为黄色、绿色、青色、蓝色、橙色。连续两种颜色的角度相差60度。



————————————————————————————————————————————————————

案例：

【OpenCV】颜色识别实例（瓶盖）

[https://blog.csdn.net/taily\_duan/article/details/51886019](https://javajgs.com/go?url=https://blog.csdn.net/taily_duan/article/details/51886019)



【OpenCV】OpenCV创建颜色识别类-class ColorDetector

[https://blog.csdn.net/Taily\_Duan/article/details/51505907](https://javajgs.com/go?url=https://blog.csdn.net/Taily_Duan/article/details/51505907)

Taily老段的微信公众号，欢迎交流学习

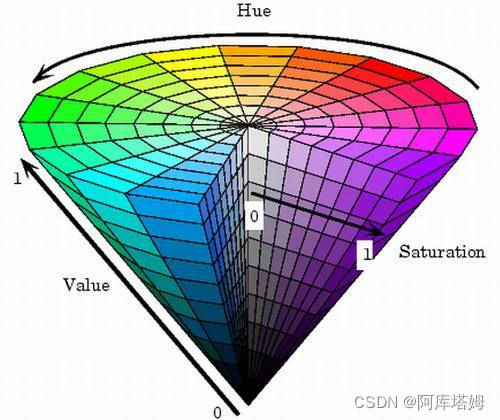


[https://blog.csdn.net/taily\_duan/article/details/81214815](https://javajgs.com/go?url=https://blog.csdn.net/taily_duan/article/details/81214815)

# 利用opencv-python的hsv空间进行图像提取

## HSV颜色空间简介

HSV(Hue, Saturation, Value)是根据颜色的直观特性由A. R. Smith在1978年创建的一种颜色空间, 也称六角锥体模型(Hexcone Model)。

这个模型中颜色的参数分别是:色调(H)，饱和度(S)，明度(V)。  
色调可以表示颜色的种类，它是一个角度，范围是0-180°，0对应red，60对应green，120对应blue。  
饱和度表示该颜色接近光谱色的程度，范围是0-255，当其为0时，只能显示黑色-白色。其逐渐增大时，颜色逐渐接近光谱色。  
明度也可以理解为亮度，范围是0-255，当其为0时，为纯黑色。逐渐增大时，可逐渐显示出颜色。  
（以上三个范围在不同的地方有不同的定义）  
如下图可以形象的理解：  
[](https://img-blog.csdnimg.cn/f4d1eb871eda4ba5a5faee396384dcee.png?x-oss-process=image/watermark,type_d3F5LXplbmhlaQ,shadow_50,text_Q1NETiBA6Zi_5bqT5aGU5aeG,size_17,color_FFFFFF,t_70,g_se,x_16#pic_center)

## 图像提取的例子

### 1.识别按钮的简单例子

设想有一工业机器人，现任务是将一墙壁上的某一按钮按下，我们现在将问题最简化然后解决。  
假设墙壁为蓝色（Blue），有两个按钮分别为红色（Red），绿色（Green），现在的任务是识别出红色（Red）按钮。  
解决过程：  
第一步：导入所需库：

import cv2 as cv

import numpy as np

第二步：创建墙壁模型：

ground = np.ones((512,512,3),np.uint8)

for i in range(512):

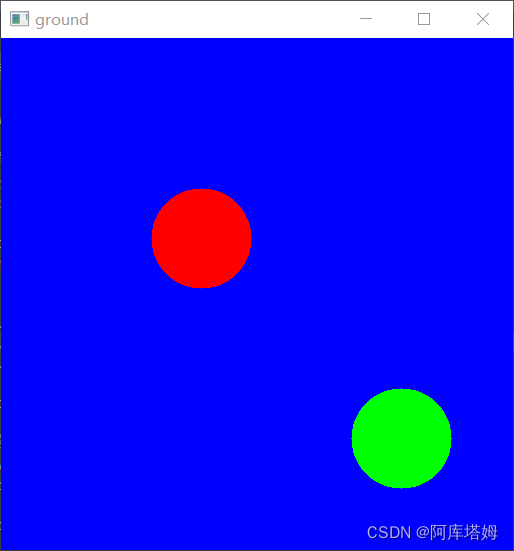
for j in range(512):

ground[i,j] = [255,0,0]

cv.circle(ground,(200,200),50,(0,0,255),-1)

cv.circle(ground,(400,400),50,(0,255,0),-1)

cv.imshow('ground',ground)

墙壁图片如下：  
[](https://img-blog.csdnimg.cn/de86bac5505b4e54b70f0153ab8a4bac.png?x-oss-process=image/watermark,type_d3F5LXplbmhlaQ,shadow_50,text_Q1NETiBA6Zi_5bqT5aGU5aeG,size_17,color_FFFFFF,t_70,g_se,x_16#pic_center)

第三步：将墙壁图片转化到HSV空间：

ground\_hsv = cv.cvtColor(ground,cv.COLOR\_BGR2HSV)

第四步：设置红色的取值范围并将红色取出：

lower = np.array([0,200,200])

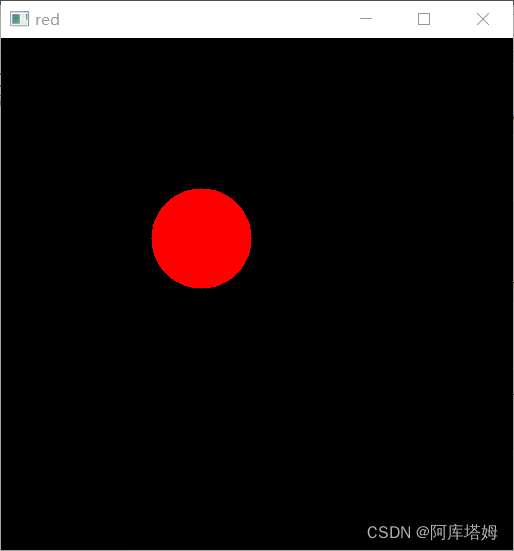
upper = np.array([20,255,255])

red\_mask = cv.inRange(ground\_hsv,lower,upper)

当待识别颜色不是标准三原色时，我们可以事先查询该颜色对应的HSV颜色，在例子2中会使用到。  
第五步：对原图像进行处理：

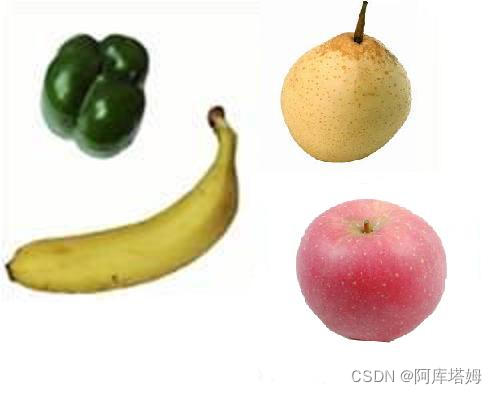
red = cv.bitwise\_and(ground,ground,mask=red\_mask)

cv.imshow('red',red)

显示的结果为：  
[](https://img-blog.csdnimg.cn/42c330addfc147778124ca92a06e8e0a.png?x-oss-process=image/watermark,type_d3F5LXplbmhlaQ,shadow_50,text_Q1NETiBA6Zi_5bqT5aGU5aeG,size_17,color_FFFFFF,t_70,g_se,x_16#pic_center)

可见已将红色按钮区域识别出

### 2.水果分类

我们的目标是在如下一张含有四种水果的图中，找出梨的位置：  
[](https://img-blog.csdnimg.cn/d0c6afed1a404243937d08501fa725b4.png?x-oss-process=image/watermark,type_d3F5LXplbmhlaQ,shadow_50,text_Q1NETiBA6Zi_5bqT5aGU5aeG,size_17,color_FFFFFF,t_70,g_se,x_16#pic_center)

代码如下

import cv2 as cv

import numpy as np

fruit = cv.imread('D:/programming/Python learn/opencv\_learn/fruit.jpg')

pear = cv.imread('D:/programming/Python learn/opencv\_learn/pear.jpg')

cv.imshow('fruit',fruit)

#找到梨的HSV参数范围的过程：

pear\_hsv = cv.cvtColor(pear,cv.COLOR\_BGR2HSV)

a = pear\_hsv.shape[0]

b = pear\_hsv.shape[1]

h = []

s = []

v = []

for i in range(int(a/2)-10,int(a/2)+9):

for j in range(int(b/2)-10,int(b/2)+9):

h.append(pear\_hsv[i,j][0])

s.append(pear\_hsv[i,j][1])

v.append(pear\_hsv[i,j][2])

H = np.array(h)

S = np.array(s)

V = np.array(v)

h\_min = np.min(H)

h\_max = np.max(H)

s\_min = np.min(S)

s\_max = np.max(S)

v\_min = np.min(V)

v\_max = np.max(V)

fruit\_hsv = cv.cvtColor(fruit,cv.COLOR\_BGR2HSV)

lower = np.array([h\_min,s\_min-5,v\_min-5])

upper = np.array([h\_max,s\_max+5,v\_max+5])

pear\_mask = cv.inRange(fruit\_hsv,lower,upper)

pear = cv.bitwise\_and(fruit,fruit,mask=pear\_mask)

cv.imshow('pear',pear)

if cv.waitKey(-1)==ord('a'):

cv.destroyAllWindows()

输出结果为：  
[](https://img-blog.csdnimg.cn/279bfaa761364fffac584941de35be3c.png?x-oss-process=image/watermark,type_d3F5LXplbmhlaQ,shadow_50,text_Q1NETiBA6Zi_5bqT5aGU5aeG,size_17,color_FFFFFF,t_70,g_se,x_16#pic_center)

[](https://img-blog.csdnimg.cn/a8d9fd7ae120405eb71e379ad0a9e144.png?x-oss-process=image/watermark,type_d3F5LXplbmhlaQ,shadow_50,text_Q1NETiBA6Zi_5bqT5aGU5aeG,size_17,color_FFFFFF,t_70,g_se,x_16#pic_center)

可以看出虽然不完全，但可以成功将梨与其他三种水果区分开并识别出。  
在本例子中，我们寻找范围的办法是：将梨的图片单独取出，查询中心区域（大小20\*20）的像素组成，用这块区域的最小值与最大值适当扩张组成区分依据。在具体应用中，还需要加以改变，如香蕉的有效区域不在中心，苹果的亮度变化较大等等。想要实现比较精准、更大难度的分类需要进一步学习，我们通过这两个例子了解HSV提取的基本思想即可。